Fall 2010 CS61C Midterm 1 Solutions

Your Name: _____Peter Perfect_____

Your TA: Andrew Michael Conor Charles

Login: cs61c-___

This exam is worth 85 points, or about 10% of your total course grade. The exam contains 7 questions.

This booklet contains 9 numbered pages including the cover page. Put all answers on these pages, please; don't hand in stray pieces of paper.

Question	Points (Minutes)	Score
1	**9 (8)**	9
2	**8 (8)**	8
3	**12 (8)**	12
4	**9 (9)**	9
5	**14 (12)**	14
6	**9 (10)**	9
7	**24 (30)**	24
Total	**85 (85)**	85

Question 1 – Potpourri (9 points / 8 minutes)

Rubric: 2 pts / multiple choice, 1 pt / true/false.

a) Suppose we have defined the C structure:

```
struct player {
      int id;
      int numGoals;
      char name[8];
};
```

Also, we declare

```
struct player players[3];
```

such that `players` starts at `0x10000000`. What is the value of `playerTwo` after:

```
struct player *playerTwo = players + 2;
```

c) 0x10000020

b) Fill in the blank using one of the choices below. The quantity of numbers that single precision floating point can represent is _____ the quantity of numbers a 32-bit two's complement integer can represent.

b) less than

c) True / False. Circle the right answer

I) **T** The most expensive memory per bit is the smallest memory in a memory hierarchy.

II) **T** The largest capacity memory is the slowest memory in a memory hierarchy.

III) **F** For a given instruction set architecture and hardware implementation, a compiler that produces fewer instructions always produces a faster program.

IV) **F** Data level parallelism is enabled by many small and independent tasks that can be spread among identical servers.

V) **T** In map-reduce processing, the reduction step must wait until it has received data from all of the mapping steps before it can start.

Question 2 Curves – (8 points / 8 minutes)

"Magic lives in curves, not angles." Mason Cooley

Rubric: 2 pts per part.

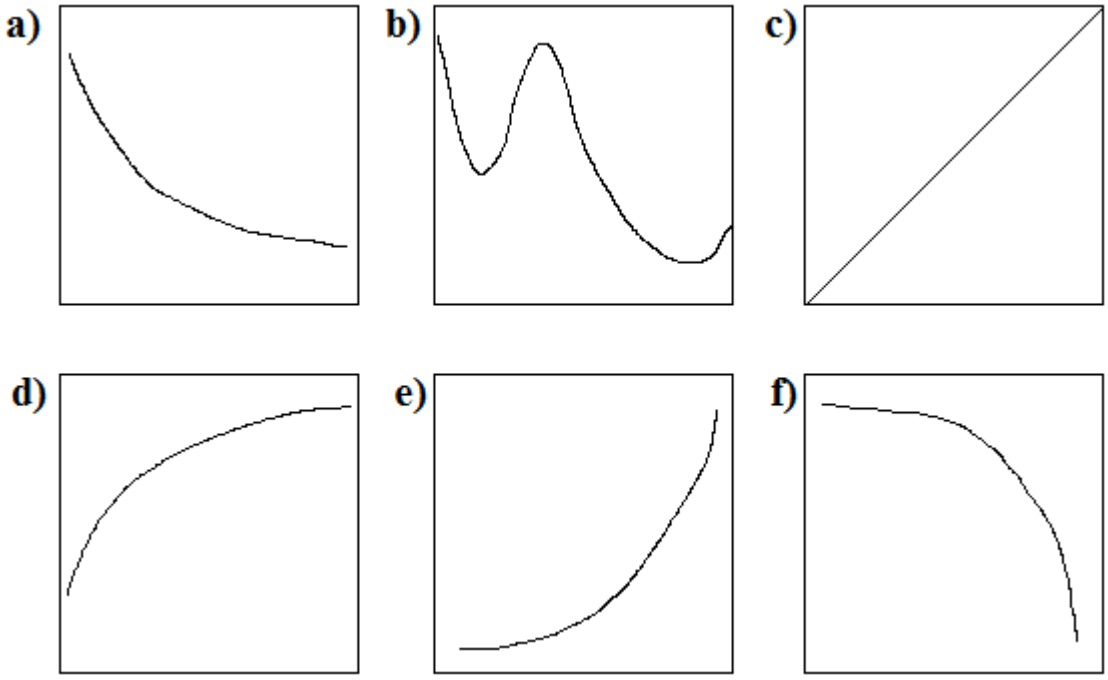
Please match each of the following descriptions with the graphs below. **Assume a linear scale for both the X and Y axes.**

I) A technology's performance near the end of its lifespan. **(time on X Axis)** **d**

II) Energy-proportional server energy usage as a function of utilization. **(utilization on X Axis)** **c**

III) Density of transistors on a chip during the 90s. **(time on X Axis)** **e**

IV) Typical server utilization histogram for servers in a datacenter. **b**



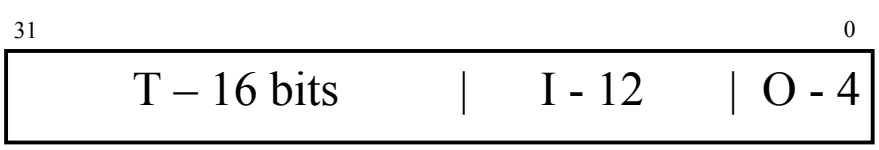
Question 3 – Hidden Treasure (12 points / 8 minutes)

"A box without hinges, key, or lid, yet golden treasure inside is hid." J. R. R. Tolkien

Rubric: 6 pts/ per part.

a) Consider a 32-bit byte address machine with a direct mapped cache. The cache is organized with 4096 blocks of four 32-bit words, each block using write back cache policy.

Draw and label the partitionings of the 32-bit memory address into the segments that are used to access the cache; Label each segment with its name and its width in bits. (shows bit numbers)



Rubric: 2 pts for tag, 2 pts for index, 2 pts for offset

- ***-1 pt for correct bit numbers but incorrect subtraction***
- ***-1 pt for tag in the middle***
- ***-3 pts for dirty and valid bits in address***

b) Including all of the cache management bits, what is the total number of bits in the cache? **Show your work, and to simplify the calculation, give your answer in kbits (1024 bits).**

128 bits / block + 16 + 2 = 146 bits / row.

4K blocks * 146 = 584 kbits

Rubric:

- ***-2 pts for subtracting tag + management bits***
- ***-1 pt if no valid bit***
- ***-1 pt if no dirty bit***
- ***-2 pts if used 32-bit tag or index in cache***
- ***-1 pt if more than one valid bit***
- ***-2 pts computed correct answer but divided by a constant***
- ***-2 pts for very bad miscalculations***

Question 4 – Shifty (9 points / 9 minutes)

"Twenty years of schoolin' / And they put you on the day shift." Bob Dylan

It was mentioned in lecture that computer designers had been fooled into thinking an arithmetic shift right (SRA) instruction is identical in effect to dividing by two.

SRA performs identically to shift right logical (SRL) for positive two's compliment numbers, but fills the leftmost bits with 1's for negative numbers (most significant bit 1). Here is an example assuming an 8 bit word

SRA by 3 on 0010 1001 => 0000 0101.
SRA by 3 on 1010 1001 => 1111 0101.

For one byte Two's Complement integers, give an example that shows SRA cannot be used to implement the C ***signed*** integer division operator by powers of two. ***You only need to use 8-bit words in your example.***

1111 1111 = -1
SRA(1111 1111, 1) = -1. Does not divide by 2.

OR:

Inconsistency in rounding schemes

0000 0101 = 5
SRA(0000 0101, 2) = 1. Rounds down.

1111 1011 = -5
SRA(1111 1011,2) = -2. Rounds up.

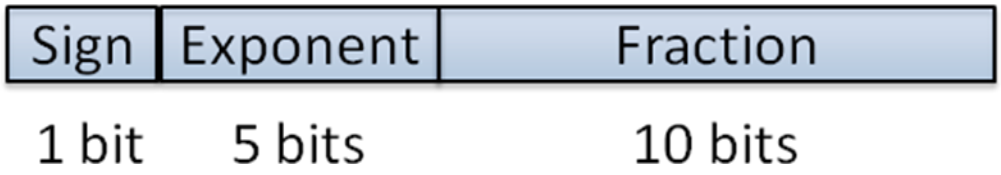
Rubric: all or nothing; common mistakes:

1. ***to work the question using sign + magnitude numbers (the question explictly stated twos complement numbers)***
2. ***to invent their own rounding scheme (the question explictly stated to use C semantics, which truncates -- rounds toward 0), and***
3. ***to come up with a counter example in which shifting right actually worked, when the goal was to show an example where it didn't. The quickest, easiest example was -1 (1111 1111 in twos complement), shifted any number of places to the right always yields -1, when it should round to 0).***

Question 5 – Halfway to Floating Point (14 points / 12 minutes)

"Anyone who thinks there's safety in numbers hasn't looked at the stock market pages." Irene Peter

(Half Precision) The recent revision of the floating-point standard added 16-bit floating-point precision, with the following format:



a) Assuming that half-precision follows the same philosophy as single and double precision, what should be the bias for this 5-bit exponent in half precision? **01111 = 15**

Rubric: 4 pts. 2 pts for a bias that would be correct for a different exponent bit width (ie 31), or putting 16.

b) What real ***decimal*** number (or symbol) do you expect each of the following binary bit patterns represent if they are half precision floating point numbers?

I) 0 00000 0000000000 **0**

II) 0 01111 0000000000 **1**

III) 1 10000 0000000000 **-2**

IV) 1 11111 0000000000 **-infinity**

V) 1 11111 1111111111 **-NAN**

Rubric: 2 pts per part, graded relative to the bias you gave in a). Special cases:

- ***-1 pt each for sign errors, capped at -3 total.***
- ***-4 pts for not recognizing zero/infty/NaN (floating point special forms). Partial***
- ***credit awarded based on derivation of these numbers according to the standard FP equation.***
- ***-2 pts for not recognizing infty/NaN, or confusing the two.***
- ***For II and III, some credit awarded if both parts made the same errors in computation, but were at least partially correct.***

Question 6 – Returning Mystery (9 points / 10 minutes)

"Mystery is at the heart of creativity. That, and surprise." Julia Cameron

What does the assembly function mystery return? Write your answer as a **binary number**. In **one short phrase**, describe how you got your answer. You may use some kind of obvious shorthand to denote long strings of ones or zeros, e.g. 16{1} to denote 16 ones in a row.

Address		**Instruction**
0x08001000	mystery:	addiu $sp, $sp, -4
0x08001004		sw $ra, 0($sp)
0x08001008		addiu $v0, $zero, 0
0x0800100c		jal inner
0x08001010		lw $ra, 0($sp)
0x08001014		addiu $sp, $sp, 4
0x08001018		jr $ra
0x0800101c	inner:	lw $v0, 4($ra)
0x08001020		jr $ra

001001 11101 11101 0000 0000 0000 0100

001001 11101 11101 13{0} 100

Binary encoding of addiu $sp, $sp, 4.

Rubric: Full credit for correct binary number if we could tell it was produced by assembling addiu $sp, $sp, 4.

Partial credit (non-exhaustive list):

- ***7 pts: assembled wrong instruction due to off-by-one or direction error in computing $ra or offset from $ra;***
- ***4 pts: binary of $ra + 4 (= 0x08001014; address loaded) w/explanation that it was computed from $ra + 4***
- ***3 pts: binary of $ra + 4, computed incorrectly (due to, e.g., off-by-one error)***
- ***2 pts: miscellaneous function of $ra with correct location for $ra***
- ***2 pts: "0" with statement that $v0 was never clobbered after "addiu $v0, $zero, 0"***
- ***1 pt: miscellaneous function of $ra with wrong location for $ra***
- ***0 pts: wrong number with no or meaningless explanation***

Question 7 – Metasyntactic Function (24 points / 30 minutes)

"Garply: A metasyntactic variable like foo, once popular among SAIL hackers."
Computing Dictionary

a) Examine the following MIPS function `garply`. Some of the instructions related to function calling conventions have been omitted. Fill in the instructions so that `garply` can be used correctly. Then assemble the 4 instructions that were given in the problem into binary MIPS machine code. Give your answers in binary, and then hexadecimal. Assume that the base address of the function `garply` is at `0x10000000`. You may use some kind of obvious shorthand to denote long strings of ones or zeros, e.g. 16{1} to denote 16 ones in a row. See examples below.

MIPS assembly	Machine Code	
	Binary	**Hexadecimal**
garply:		
addiu $sp,$sp,-4		
sw $ra, 0($sp)		
addiu $v0 $zero 0	001000 00000 00010 16{0}	0x20020000
lbu $t0 0($a0)	**100000 00100 01000 16{0}**	**0x80880000**
beq $t0 $zero end	**000100 01000 00000 14{0} 11**	**0x11000003**
addiu $a0 $a0 1	**001000 00100 00100 15{0} 1**	**0x20840001**
jal garply	**000011 26{0}**	**0x0c000000**
addiu $v0 $v0 1	001000 00010 00010 (15{0}) 1	0x20420001
end:		
lw $ra, 0($sp)		
addiu $sp,$sp,4		
jr $ra		

Rubric:
Six for the MIPS prologue/epilogue. Two points off for each major error (max of six points off). One point off for minor errors.

Eight for the binary/hex. Each assembly into binary was worth 1.5 points; most errors resulted in losing these 1.5 points, but very minor ones could result in less being taken off. each translation into hex was worth .5 points, and you got these points if you correctly translated from binary, even if your binary was wrong. We rounded up the points lost to the next integer, so if you missed two binary and one hex, you were docked 4 points. Also, if you translated the addiu as addi because our mid-exam announcement confused you, we decided you should receive full credit.

b) Give a direct translation of `garply` into C. Don't change how the function computes its answer (i.e., don't change recursion into iteration, or vice-versa, or anything like that). Your solution shouldn't take more than 3 lines of code, but we'll give you up to six. Don't forget to correctly fill in the argument and return types!

```
int  garply(char* str) {
        if(*str==0)
                return 0;
        return garply(str+1)+1;
}
```

Rubric: Eight points for the translation into C. We docked two points per major error, and one point for very minor errors (like using postfix ++ instead of prefix).

c) strlen. Finds the length of a string.

Rubric: Two points for correctly stating what garply does. One point for something sort of on the right track, zero for something completely off. In general we were a lot more generous with this if your code showed that you understood what was going on. If you said something really vague like "counts the number of elements until the first zero one" but your code dealt with ints rather than chars, you were liable to lose a point.